REMARKS

Applicant respectfully requests reconsideration and allowance of the subject application. Claim 44 has been amended. Claims 1-44 are pending in this application.

Claim 44 has been amended to correct a typographical error. Claim 44 has not been amended to overcome, and should not be interpreted as having been made to overcome, any rejection of claim 44.

35 U.S.C. § 102

Claims 1-44. stand rejected under 35 U.S.C. §102(e) as being unpatentable over U.S. Patent No. 6,735,311 to Rump et al. (hereinafter "Rump"). Applicant respectfully submits that claims 1-44 are not anticipated by Rump.

Rump is directed to encryption and decryption of multi-media data (see, Title). As discussed in the abstract of Rump, Rump discusses a method for ciphering multimedia includes the entering of a ciphering index in a definition data block of the multimedia data, this index pointing to a ciphering algorithm which is to be used. In response to the ciphering index in the definition data block one of a plurality of ciphering algorithms is selected. The multimedia data are ciphered using the selected ciphering algorithm. Various additional entries in the definition data block which is assigned to the multimedia data permit the clearing or enabling of a deciphering device, rapid access to a database of ciphered multimedia data and a customer- and data-specific use of the multimedia data while taking copyright aspects into account.

The definition data block consists of two parts, namely a fixed part 10, which is shown in Fig. 1, and a variable part 30, which is shown in Fig. 2 (see, Fig. 1, Fig. 2, and col. 5, lines 16-18). Each of these two parts includes various entries, as shown in the tables of Fig. 1 and Fig. 2. In each table of Fig. 1 and Fig. 2, the last column of the table indicates whether the entry is ciphered or not (see, col. 5, lines 30-32, and col. 7, lines 11-13).

In contrast, claim 1 recites in part:

> identifying a plurality of modules in a software program, wherein each module includes a plurality of blocks and wherein the plurality of modules includes checker modules;
>
> for each of the plurality of modules,
>
> generating an original checkpoint value, and
>
> incorporating the original checkpoint value into a checker module; and
>
> for each of the checker modules,
>
> generating a new checkpoint value after the original checkpoint value has been incorporated into the checker module, and
>
> determining a new block to add to the checker module to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker module equals the original checkpoint value for the checker module.

As can be seen in claim 1, an original checkpoint value is generated for each of a plurality of modules in a software program, and this original checkpoint value is incorporated into a checker module. A new checkpoint value is also generated for each of the checker modules after the original checkpoint value has been incorporated into the checker module. Thus, for a checker module, two different checkpoint values are generated: an original checkpoint value and a new checkpoint value. Applicant respectfully submits

that Rump does not disclose generating two such checkpoint values for a checker module as recited in claim 1.

It appears from the July 2 Office Action at ¶3, pp. 2-3, that the checksum of Rump is being relied on as disclosing the checker modules. See, for example, ¶3, p. 2, of the July 2 Office Action stating that "Rump et al. discloses . . . wherein the plurality of modules includes checker modules (i.e. checksum), each data block contains a checksum . . . ."

Rump describes an entry 20 in the definition data block that is called the checksum (see, col. 6, lines 38-42). This checksum consists of an MD5 fingerprint of the definition data block (10 and 30 of Figs. 1 and 2), or may contain both the definition data block and also a specified number of multimedia data to be ciphered to which the described definition data block is assigned (see, col. 6, lines 51-57).

Applicant respectfully submits that there are no two checkpoint values generated for the checksum discussed in Rump. The checksum of Rump is itself a checksum value (an MD5 fingerprint), but there are no two checkpoint values generated for this checksum value of Rump. In order for the checksum of Rump to satisfy the language of claim 1, an original checkpoint value would have to be generated for the checksum, that original checkpoint value would have to be incorporated into the checksum, and then a new checkpoint value would have to be generated for the checksum. As there is no discussion or mention of such behavior in Rump, Applicant respectfully submits that Rump cannot disclose claim 1.

Additionally, Rump discusses a free index. The free index contains a serial number which is a 32-bit long serial number which identifies the multimedia data, and user data which is the first 12 bytes of the MD5 fingerprint of the first data of the multimedia data block from Amount to Step minus Amount (see, col. 8, lines 37-44). This free index of Rump is a serial number and the first 12 bytes of the MD5 fingerprint of the first data of the multimedia data block, not a checkpoint value generated for the checksum of Rump.

As Rump does not disclose generating two such checkpoint values for a checker module as recited in claim 1, Applicant respectfully submits that claim 1 is not anticipated by Rump.

Furthermore, as can be seen in claim 1, the original checkpoint value is incorporated into a checker module and a new checkpoint value is generated for the checker module after the original checkpoint value has been incorporated into the checker module. Additionally, a new block is determined to add to the checker module to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker module equals the original checkpoint value for the checker module. Applicant respectfully submits that Rump does not disclose any such new block to add to the checker module to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker module equals the original checkpoint value for the checker module.

It appears from the July 2 Office Action at ¶3, pp. 2-3, that the free index field of Rump is being relied on as disclosing the determining of this new block.

See, for example, ¶3, p. 3, of the July 2 Office Action stating that Rump discloses "determining a new block to add to the checker module to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker module equals the original checkpoint value for the checker module (see col. 6, lines 61-67, col. 8, lines 25-61), Rump discloses this because Rump discloses a free index field that is added to a checker module . . . .".

However, the free index of Rump, as discussed above, is a serial number and the first 12 bytes of the MD5 fingerprint of the first data of the multimedia data block. There is no discussion or mention of this serial number or first 12 bytes of the MD5 fingerprint offsetting the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker module equals the original checkpoint value for the checker module as recited in claim 1. Applicant respectfully submits that nowhere in Rump is there any discussion or mention of determining a new block based in any way on a subsequently generated checkpoint value equaling an original checkpoint value.

As Rump does not disclose any such new block to add to the checker module to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker module equals the original checkpoint value for the checker module as recited in claim 1, Applicant respectfully submits that claim 1 is not anticipated by Rump.

For at least these reasons, Applicant respectfully submits that claim 1 is allowable over Rump.

Given that claims 2-9 depend from claim 1, Applicant respectfully submits that claims 2-9 are likewise allowable over Rump for at least the reasons discussed above with respect to claim 1.

With respect to claim 10, claim 10 recites in part:

> identifying a plurality of segments in an object; and
> applying cyclic integrity verification to the object based on the plurality of segments.

Applicant respectfully submits that Rump does not disclose applying cyclic integrity verification to the object based on the plurality of segments as recited in claim 10.

As discussed in Applicant's specification at, for example, page 14, line 23 – page 15, line 6, and page 18, lines 3-14, cycles of integrity verification can be created. An example of such a cycle is segment A verifying the integrity of segment B, which in turn verifies the integrity of segment A. Such cyclic integrity verification can be troublesome, because generating a checkpoint value for segment A and storing it in segment B would change the checkpoint value that is generated for segment B and stored in segment A, which would then change the checkpoint value for segment A, and so on.

Rump at col. 6, lines 38-60 is cited in the July 2 Office Action as disclosing the method of claim 10. The cited portion of Rump discusses the checksum of the definition data block. This checksum consists of an MD5 fingerprint of the definition data block (10 and 30 of Figs. 1 and 2), or may contain both the definition data block and also a specified number of multimedia data to be ciphered to which the described definition data block is assigned (see, col. 6, lines 51-57). However, there is nothing cyclic about this

checksum of Rump. Rather, the checksum is simply a value that is generated as described in Rump and stored in the definition data block. Rump does not discuss or mention any cycles, and Rump does not discuss or mention that the checksum creates any cycles involving the checksum or the definition data block. As there is no discussion or mention of such cycles in Rump, Applicant respectfully submits that Rump cannot disclose cyclic integrity verification, much less applying cyclic integrity verification to the object based on the plurality of segments as recited in claim 10.

For at least these reasons, Applicant respectfully submits that claim 10 is allowable over Rump.

With respect to claim 11, claim 11 depends from claim 10 and Applicant respectfully submits that claim 11 is allowable over Rump at least because of its dependency on claim 10. Furthermore, Applicant respectfully submits that, similar to the discussion above regarding claim 1, Rump does not disclose generating two checkpoint values for a checker segment as recited in claim 11, and does not disclose an additional block to be added to the checker segment to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker segment equals the original checkpoint value for the checker segment as recited in claim 11. For at least these reasons, Applicant respectfully submits that claim 11 is allowable over Rump.

Given that claims 12-14 depend from claim 10, Applicant respectfully submits that claims 12-14 are likewise allowable over Rump for at least the reasons discussed above with respect to claim 10.

With respect to claim 15, Applicant respectfully submits that, similar to the discussion above regarding claim 1, Rump does not disclose modifying each of the plurality of segments so that the addition of the checkpoint value to the segment is offset and the checkpoint value for the segment remains the same as recited in claim 15. For at least these reasons, Applicant respectfully submits that claim 15 is allowable over Rump.

Given that claims 16-24 depend from claim 15, Applicant respectfully submits that claims 16-24 are likewise allowable over Rump for at least the reasons discussed above with respect to claim 15.

With respect to claim 25, Applicant respectfully submits that, similar to the discussion above regarding claim 1, Rump does not disclose adding an offset value to the second segment so that a newly calculated verification value for the second segment equals the original verification value as recited in claim 25. For at least these reasons, Applicant respectfully submits that claim 25 is allowable over Rump.

Given that claims 26-37 depend from claim 25, Applicant respectfully submits that claims 26-37 are likewise allowable over Rump for at least the reasons discussed above with respect to claim 25.

With respect to claim 38, Applicant respectfully submits that, similar to the discussion above regarding claim 10, Rump does not disclose the plurality of segments further include a plurality of checkpoints that identify a circular ordering of verifying the integrity of the segments as recited in claim 38. For at least these reasons, Applicant respectfully submits that claim 38 is allowable over Rump.

With respect to claim 44, Applicant respectfully submits that, similar to the discussion above regarding claim 10, Rump does not disclose a production server to apply [illegible]

Given that claims 39-40 depend from claim 38, Applicant respectfully submits that claims 39-40 are likewise allowable over Rump for at least the reasons discussed above with respect to claim 38.

With respect to claim 41, Applicant respectfully submits that, similar to the discussion above regarding claim 10, Rump does not disclose the production server being configured to parse the original program into a plurality of segments and apply cyclic integrity verification to the plurality of segments as recited in claim 41. For at least these reasons, Applicant respectfully submits that claim 41 is allowable over Rump.

With respect to claim 42, claim 42 depends from claim 41 and Applicant respectfully submits that claim 42 is allowable over Rump at least because of its dependency on claim 41. Furthermore, Applicant respectfully submits that, similar to the discussion above regarding claim 1, Rump does not disclose generating two checkpoint values for a checker segment as recited in claim 42, and does not disclose an additional block to be added to the checker segment to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker segment equals the original checkpoint value for the checker segment as recited in claim 42. For at least these reasons, Applicant respectfully submits that claim 42 is allowable over Rump.

Given that claim 43 depends from claim 41, Applicant respectfully submits that claim 43 is likewise allowable over Rump for at least the reasons discussed above with respect to claim 41.